1986-12

A missile flyout model for ISEAS

Antonio, Dennis D.

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A MISSILE FLYOUT MODEL FOR ISEAS

by

Dennis D. Antonio

December 1986

Thesis Advisor: Robert E. Ball

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION: Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable): 67	7a NAME OF MONITORING ORGANIZATION: Naval Postrgraduate School
6c ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5100	7b ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5100
8a NAME OF FUNDING/SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)

A MISSILE FLYOUT MODEL FOR ISEAS

12 PERSONAL AUTHOR(S)

Dennis D. Antonio

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1986 December	59

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	PC - Based Computer Simulation of Air Defense (ISEAS) for Surface to Air Missiles

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

A computer model of the launch and flyout of a shipboard surface to air missile toward an attacking aircraft or anti-ship missile has been developed for the Interactive Simulation of Engagements at Sea (ISEAS) program. The model, written in the C language, is based upon the MICE II Fortran program. The flyout model can be used to asses missile system performance against an air target. A complete discription of the missile model, with flow charts and a C program listing, is included.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL: Robert E. Ball	22b TELEPHONE (Include Area Code): (408) 646-2885; 22c OFFICE SYMBOL: 67Bp

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Approved for public release; distribution is unlimited

A Missile Flyout Model for ISEAS

by

Dennis D. Antonio
Lieutenant, United States Navy
B.S., United States Naval Academy, 1977

Submitted in partial fulfillment of the
requirements for the degress of

MASTER OF SCIENCE IN ENGINEERING SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1986

ABSTRACT

A computer model of the launch and flyout of a shipboard surface to air missile toward an attacking aircraft or anti-ship missile has been developed for the Interactive Simulation of Engagements at Sea (ISEAS) program. The model, written in the C language, is based upon the MICE II Fortran program. The flyout model can be used to assess missile system performance against an air target. A complete discription of the missile model, with flow charts and a C program listing, is included.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

ISEAS (Interactive Simulation of Engagements At Sea) is a computer program currently under developement at the Naval Postgraduate School to simulate interactions between a ship loaded with surface-to-air missile systems and an attacking aircraft or anti-ship missile. The portion of ISEAS described in this thesis is a translation of the Fortran program MICE II [Ref. 1:pp 4-2Ø] for the flyout of a surface to air missile to the C language. Three different missile systems are simulated. The C program includes coordinate transformations, launcher bearing and elevation calculations, the proportional navigation equation for missile guidance, and the missile's equations of motion. The program contains a description of the missile engagements with an air target and provides a basis for the evaluation of surface-to-air missile launching opportunities and subsequent missile performance under realistic encounter conditions. This missile program provides the basic framework from which further simulations of increased complexity and sophistication can be easily implemented.

II. PROPORTIONAL NAVIGATION

Guided missiles are navigated along a path that leads to an intercept with the air target. Proportional Navigation is one type of navigation in which the rate of change of the missile heading is made proportional to rate of change of the line of sight between the missile and the target. See Figure 1 for the Proportional Navigation Guidance

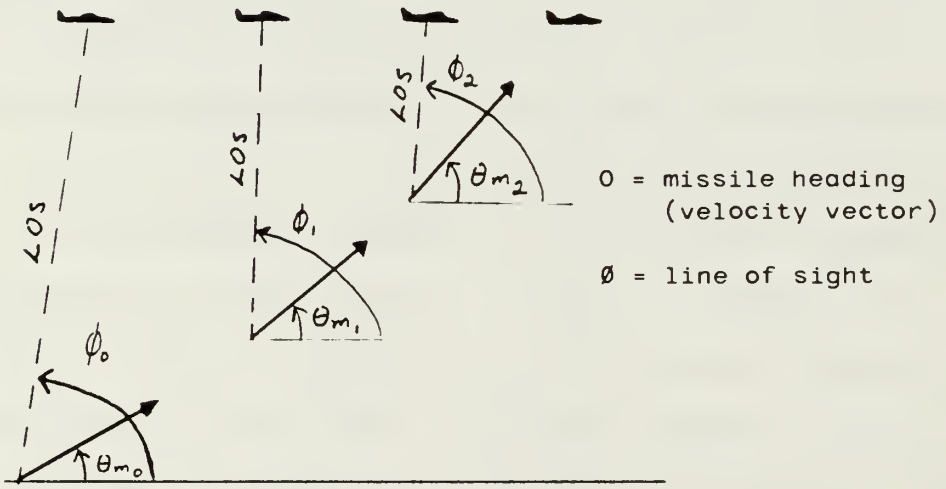

Figure 1. Proportional Navigation Guidance Trajectory

Trajectory. The equation governing Proportional Navigation is

$$\dot{\theta} = k * \dot{\phi} \qquad (1)$$

where

$\dot{\theta}$ = time rate of change of missile heading (velocity vector)

$\dot{\phi}$ = time rate of change of line of sight

k = Proportional Navigation constant

The missile rate of turn ($\dot{\theta}$) is a fixed proportion of the rate of turn of the line of sight ($\dot{\emptyset}$), but the proportional constant itself (k) can be variable during the flight. This method is implemented in MICE II. The input base value k_o, is multiplied by the ratio of the closing velocity to the missile velocity. Consequently, due to the changes in velocities, the effective value of the Proportional Navigation constant is a variable during the missile flight. There are two special cases of Proportional Navigation, Pursuit and Constant Bearing or lead angle. The Pursuit Navigation course can be obtained from Proportional Navigation by setting k = 1, and the Constant Bearing course can be obtained from Proportional Navigation by letting k be very large.

A missile traveling a Proportional Navigation course is usually launched at a lead angle which corresponds to the Constant Bearing path. (Note that a pure Pursuit course is actually the unique case of a Proportional Navigation path with a zero lead angle and a navigation constant of one). Proportional Navigation paths are less curved (fewer g's on the missile) than the Pursuit course, but may be more curved than the Constant Bearing course. The advantage of the Proportional Navigation course is that it provides a practical method of causing an intercept with a manuvering target. Suitable values of the navigation constant are between three and six. [Ref. 2:p. 67]

Figure 2 shows the two-dimensional geometry for a Proportional Navigation course. The missile and target are a distance r apart. The line of sight between the missile and the target is the angle $\emptyset$ relative to the global reference (X,Z) for all times of flight. The line of sight is at an angle α_m relative to the missile velocity vector,

and the line of sight is at an angle α_t relative to the target velocity vector. The relative closing velocity between the missile and the target ($\dot{r}$) can be written as [Ref. 2:p. 68]

$$\dot{r} = V_t^* \cos\alpha_t - V_m^* \cos\alpha_m \qquad (2)$$

where $V_t^* \cos\alpha_t$ = velocity components of the target along the line of sight

$V_m^* \cos\alpha_m$ = velocity components of the missile along the line of sight

The rotation rate of the line of sight can be written in the form [Ref. 2:p. 68]

$$r\dot{\phi} = -V_t^* \sin\alpha_t + V_m^* \sin\alpha_m \qquad (3)$$

where $V_t^* \sin\alpha_t$ = velocity vector component of the target normal to the line of sight

$V_m^* \sin\alpha_m$ = velocity vector component of the missile normal to the line of sight

$\dot{\phi}$ = rate of change of the line of sight angle

Equations (2) and (3) are the basic equations of motion for Proportional Navigation.

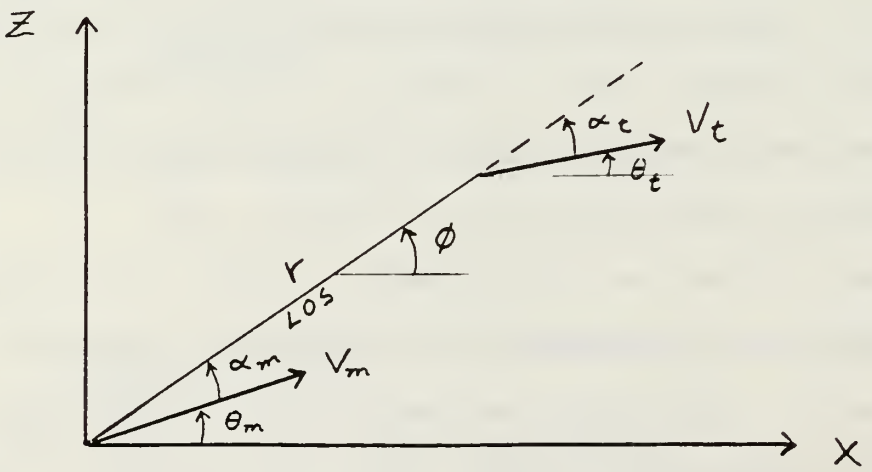

Figure 2. Proportional Navigation Geometry

III. THE MISSILE PROGRAM

A. GENERIC MISSILES

Simulations of three types of surface-to-air missiles; Long Range, Medium Range, Short Range, are developed for ISEAS. The general default missile parameters for each missile type are indicated below:

	Long Range	Medium Range	Short Range
wt (kg)	1Ø66	581	231
body dia (m)	Ø.35	Ø.35	Ø.2Ø
length (m)	7.98	4.47	3.7Ø
wing span (m)	Ø.91	Ø.91	1.ØØ
reference area (sq. m)	8.78	4.92	2.33
range (km)	121	48	2Ø
max alt (km)	2Ø	2Ø	2Ø
max flight time (sec)	2ØØ	8Ø	4Ø
avg spd (mach)	2.5	2.5	2.5
thrust (kN)	4ØØ	4ØØ	2ØØ
initial velocity (m/sec)	67Ø	67Ø	67Ø
missile burn rate (kg/sec)	3.54	4.93	4.67
drag coeff	.2	.2	.2

The user will have the option of changing any of the missile parameters listed above, but after each missile firing run, ISEAS will return to the original missile parameters.

B. ENCOUNTER TERMINATION

1. Flight Termination

Missile flight will be terminated if any of the following conditions are encountered:

a. Missile range exceeds maximum range.

b. Missile altitude exceeds maximum altitude.

c. Missile time-of-flight exceeds maximum time of flight.

2. Guidance Termination

The ability to terminate missile guidance due to a target tracker break-lock is not incorporated.

C. MISSILE POSITION IN FLIGHT

1. Global Coordinate System

The global cartesian coordinate system is used to define the target location. MICE II specified no reference point for its global coordinate, so for the ISEAS missile module the global coordinate is attached on sea surface with the Y vector to the north, the X vector to the east, and the Z vector perpendicular to X and Y vectors of the X,Y,Z coordinates (See Figure 3).

2. Missile Stability Coordinate System

The missile stability axis system is used to orient the missile in the global space, and its origin is at the missile's center of gravity. The x axis is along the missile velocity vector (with

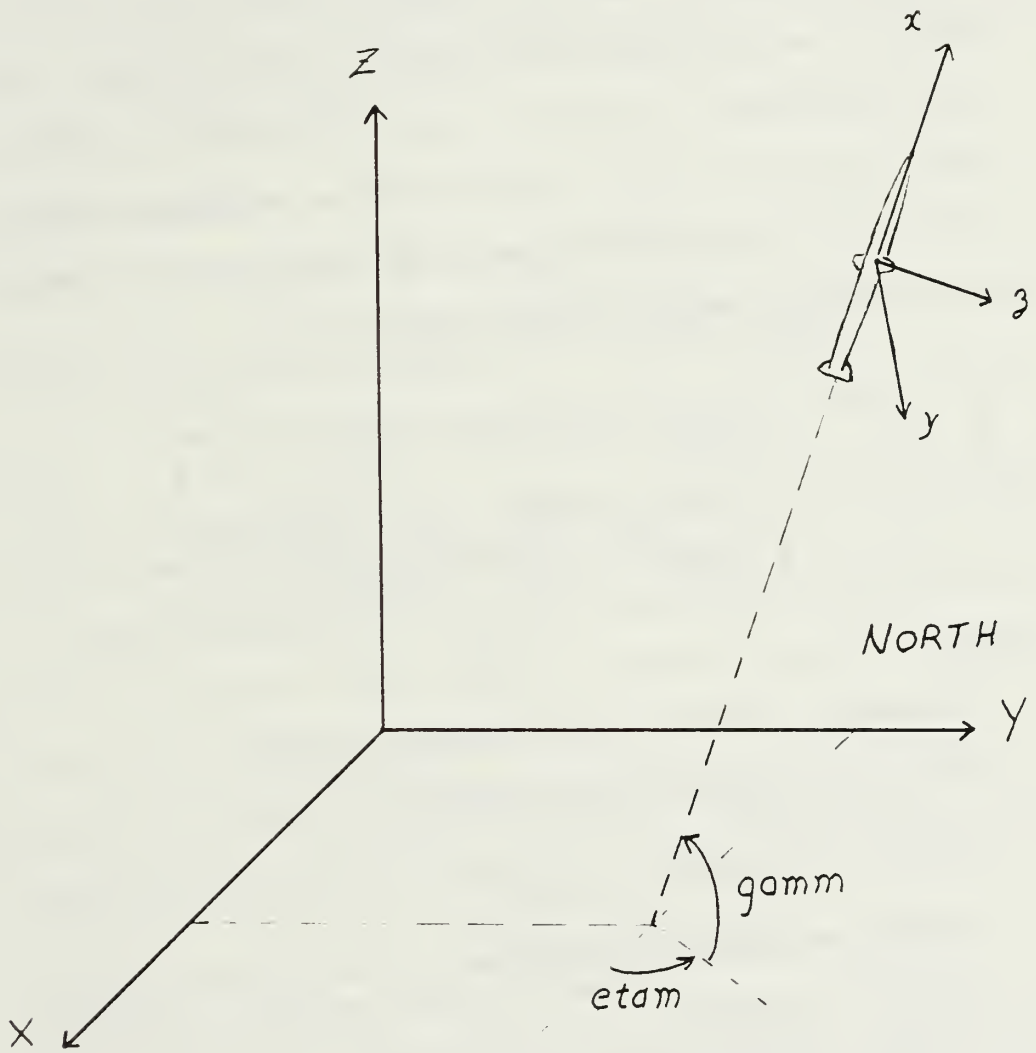

Figure 3. Coordinate Systems

positive forward), the y axis is along the right wing, and the z axis is perpendicular to the x and y axis (with positive downward). See Figure 3.

The coordinates X,Y,Z locate the the missile's center of gravity with respect to the global coordinates. The angles gamm and etam, shown in Figure 3, are the elevation and azimuth angles of the missile's velocity vector with respect to the horizontal plane of the global coordinates. Once the missile orientation is defined in the global coordinate system, transformation to the missile stability coordinate system can be performed using the following direction cosine matrix in the C language format. [Ref. 1:p. 4]

$$\begin{bmatrix} \bar{i}_x \\ \bar{i}_y \\ \bar{i}_z \end{bmatrix} = \begin{bmatrix} T[0][0] & T[0][1] & T[0][2] \\ T[1][0] & T[1][1] & T[1][2] \\ T[2][0] & T[2][1] & T[2][2] \end{bmatrix} \begin{bmatrix} \bar{I}_X \\ \bar{I}_Y \\ \bar{I}_Z \end{bmatrix} \qquad (4)$$

where

$\bar{I}_X, \bar{I}_Y, \bar{I}_Z$ = the unit global coordinate vectors

$\bar{i}_x, \bar{i}_y, \bar{i}_z$ = the unit missile coordinate vectors

T[0][0] = cos(gamm)*cos(etam)

T[0][1] = cos(gamm)*sin(etam)

T[0][2] = sin(gamm)

T[1][0] = sin(etam)

T[1][1] = cos(etam)

T[1][2] = 0

T[2][Ø] = -sin(gamm)*cos(etam)

T[2][1] = -sin(gamm)*sin(etam)

T[2][2] = cos(gamm)

For example,

$$\bar{i}_x = \cos(gamm)*\cos(etam)*\bar{I}_x + \cos(gamm)*\sin(etam)*\bar{I}_y + \sin(gamm)*\bar{I}_z$$

is the unit vector of the missile stability coordinate in the direction of the missile velocity vector.

D. LAUNCHER PARAMETERS

A missile launch requires the following conditions:

1. Line of sight range (from the missile to target) must be less than or equal to the maximum target lock-on range of seeker.

2. Line of sight range must be less than the maximum missile launch range.

3. Distance to the target after total missile flight time is less than or equal to the maximum missile range.

MICE II missile launchers are stationary and have no launcher bearing constraints. Since the ISEAS module's missile launcher is located aboard ship, and realistic shipboard launch conditions are desired, the following limitations were added to the missile launcher requirements.

4. Launcher bearings must not exceed 135 degrees port or starboard of ship's head.

5. Launcher elevation must be less than or equal to 9Ø degrees. This will permit a vertical launch if desired.

If any of the above requirements are not satisfied, missile launching will not be permitted, and target tracking and intercept solution calculations will continue until all conditions are met. If launch conditions are satisfied, missile launch will be permitted.

The m_lchr function computes the launch angles (Azi and Elev) based on current ship and calculated target intercept positions. See Figure 4 for the m_lchr block diagram. Since the launcher is located on the ship, and the ship is in motion, launch angles will be relative to ship's heading. The launch angles computation is interpreted from MICE II but with added launcher constraints as discussed above. The launch angle equations are [Ref. 1:p. 11]

$$Azi = \arctan\left[\frac{YTINT - YMSL}{XTINT - XMSL}\right] \qquad (5)$$

$$Elev = \arctan\left[\frac{ZTINT - ZMSL}{\sqrt{(XTINT - XMSL) + (YTINT - YMSL)}}\right] \qquad (6)$$

where

XMSL, YMSL, ZMSL = the missile's initial position

XTINT, YTINT, ZTINT = the target's position at time of intercept (provided by others).

If the azimuth or elevation limits are exceeded, m_lchr will return to the main function in ISEAS for recomputation of intercept solutions or possibly for the user to maneuver the ship for better launch angles. If Azi and Elev are within the azimuth and elevation limits, the program will next check if the target is within range. If out of range, m_lchr will return to the main function and the radar will continue tracking the target. If the target is within range, m_lchr will return

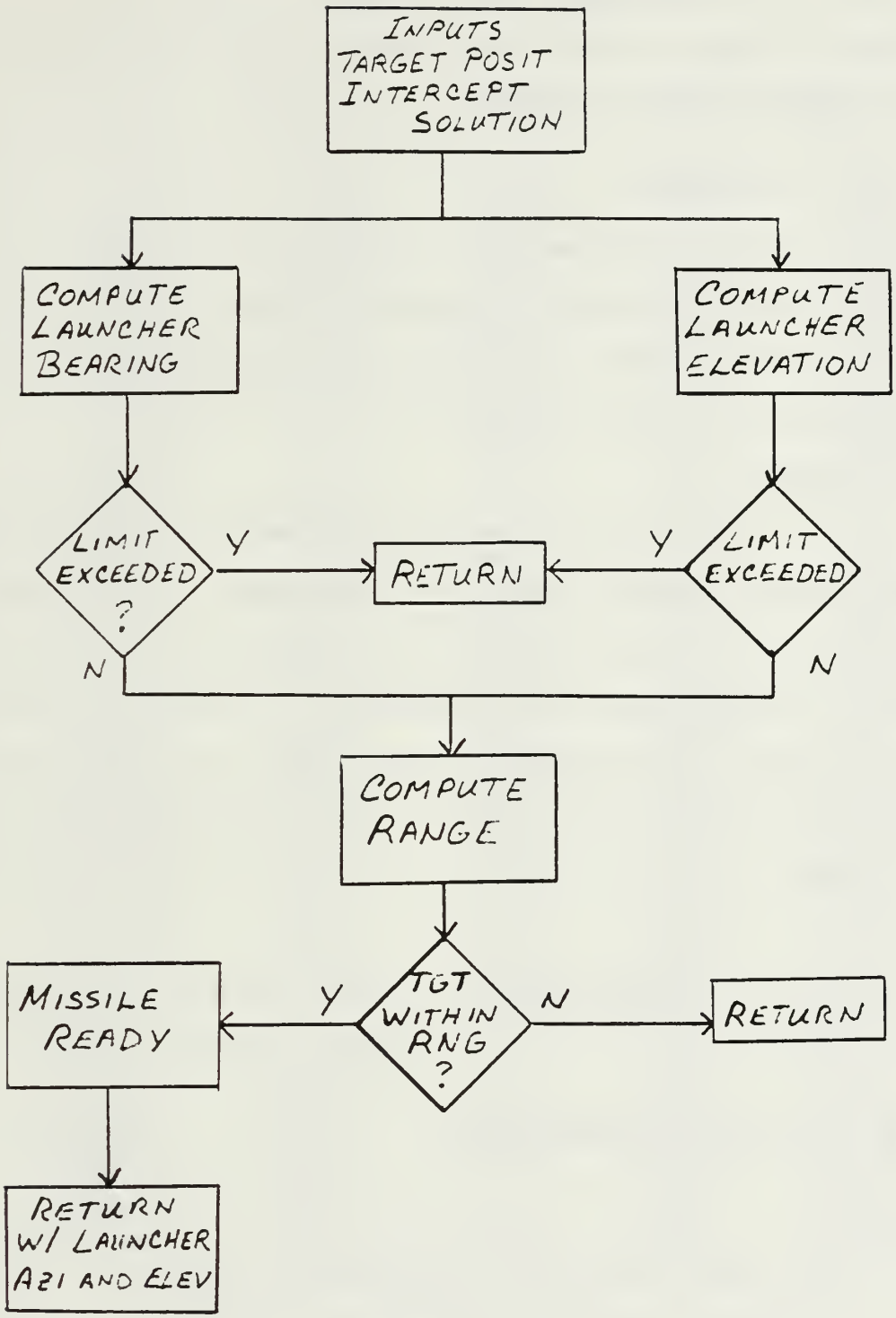

Figure 4. M_Lchr Flow Diagram

with the calculated azimuth and elevation values. This will be converted to angles in the global coordinate system and will be the initial gamm and etam of the missile.

E. MISSILE FLYOUT PHASE

There are four functions that m_flyout calls to compute the missile's linear acceleration and angular velocities. These functions are the m_atm(), m_cna(), m_thrust(), and m_guide(). M_flyout is discussed after these four functions.

1. M Atm( )

The m_atm function determines the atmospheric densities and temperatures as a function of missile altitude (ZMSL). MICE II employs a table look-up routine to determine the atmospheric temperature and density. The temperatures and air densities in ISEAS are approximated by the following linear regression equations [Ref. 3:p. 290] based on the atmospheric data of Table 1 [Ref 1:p. 47]. See Figure 5 for m_atm flow diagram.

$$\begin{aligned} &\text{temp} = 287.25 - 0.00624185 * \text{ZMSL}, && \text{for ZMSL} < 11500 \text{ m} \\ &\text{temp} = 216.6, && \text{for ZMSL} > 11500 \text{ m} \end{aligned} \tag{7}$$

$$\text{rho} = 1.0177572 - 0.000048539 * \text{ZMSL} \tag{8}$$

where

temp = temperature of air at altitude ZMSL [Kelvin]

rho = density of air at altitude ZMSL [kg/cu. m]

TABLE 1. ATMOSPHERIC DATA

ALTITUDE (FT)	TEMPERATURE (DEG K)	PRESSURE (LB/SQ FT)	DENSITY (SLUG/CU FT)	SPEED SOUND (FT/SEC)
-2000.00	292.12	2273.72	.002519	1124.09
0.00	289.16	2116.22	.002377	1116.44
2000.00	294.20	1967.69	.002241	1108.74
4000.00	230.24	1827.75	.002111	1100.99
6000.00	276.29	1696.00	.001987	1093.18
8000.00	272.32	1572.07	.001869	1085.32
10000.00	268.36	1455.60	.001756	1077.40
12000.00	264.40	1346.23	.001648	1069.42
14000.00	260.44	1243.64	.001546	1061.39
16000.00	256.49	1147.49	.001449	1053.30
18000.00	252.53	1057.47	.001355	1045.14
20000.00	248.57	973.27	.001267	1036.93
22000.00	244.62	894.59	.001184	1028.65
24000.00	240.67	821.16	.001104	1020.30
26000.00	236.71	752.71	.001029	1011.88
28000.00	232.75	688.96	.000958	1003.40
30000.00	228.81	629.66	.000891	994.85
35000.00	218.93	493.34	.000739	973.14
40000.00	216.66	393.12	.000587	968.08
45000.00	216.66	309.45	.000462	968.08
50000.00	216.66	243.61	.000364	968.08
55000.00	216.66	191.80	.000287	968.08
60000.00	216.66	151.03	.000226	968.08
65000.00	216.66	118.93	.000178	968.08
70000.00	216.66	93.67	.000140	968.08
75000.00	216.66	73.78	.000110	968.08
80000.00	216.66	58.13	.000087	968.08
85000.00	219.07	45.83	.000068	973.44
90000.00	223.60	36.29	.000053	983.46
95000.00	228.13	28.88	.000041	993.38
100000.00	232.66	23.09	.000032	1003.19
105000.00	237.19	18.54	.000025	1012.91
110000.00	241.72	14.95	.000020	1022.52
115000.00	246.24	12.10	.000016	1032.05
120000.00	250.76	9.84	.000013	1041.48
125000.00	255.28	8.03	.000010	1050.82
130000.00	259.80	6.57	.000009	1060.07
135000.00	264.31	5.40	.000007	1069.24
140000.00	268.82	4.46	.000005	1078.33
145000.00	273.33	3.69	.000004	1087.34
150000.00	277.84	3.06	.000004	1096.27
155000.00	282.35	2.55	.000003	1105.12
160000.00	282.66	2.12	.000002	1105.74
165000.00	282.66	1.77	.000002	1105.74
170000.00	282.66	1.48	.000002	1105.74
175000.00	282.66	1.23	.000001	1105.74
180000.00	276.38	1.03	.000001	1093.39
185000.00	269.65	.85	.000001	1079.98
190000.00	262.91	.70	.000001	1066.41
195000.00	256.19	.59	.000001	1052.66
200000.00	249.45	.47	.000001	1038.74
205000.00	242.72	.38	.000001	1024.64

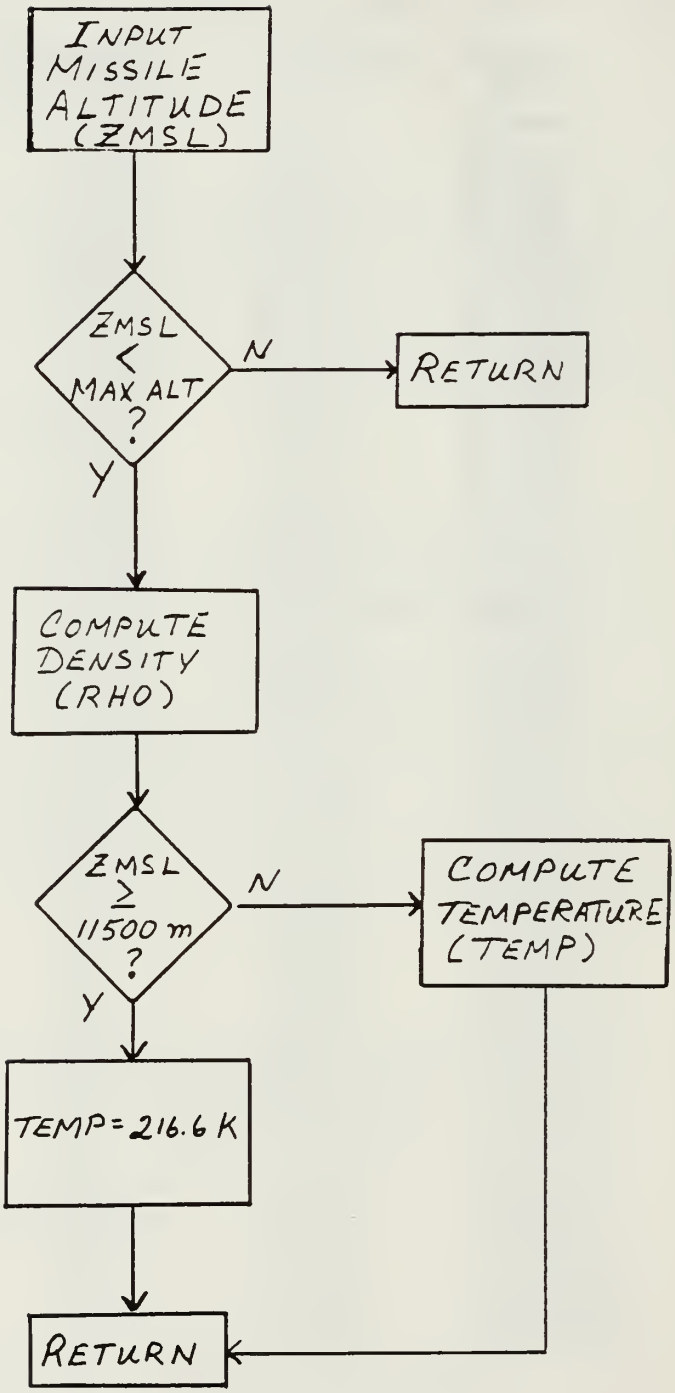

Figure 5. M_Atm Flow Diagram

2. M_Cna()

The m_cna function computes the slope of the curve that relates the lift on the missile to the angle of attack (the lift curve slope, cna) for all missiles [Ref. 2:p. 295]. MICE II used a table look-up for cna. In ISEAS, a different method is used.

The lift curve slope is dependent upon the missile altitude. The sonic velocity at altitude is computed using

$$accel = sqrt(gamma * g * R * temp) \quad (9)$$

where

gamma = 1.4

g = gravity const. [kg-m/N-sq. sec]

R = 287 [N-m/kg-Kelvin]

The missile mach number is determined by

$$mach = vmsl / accel \quad (10)$$

where

vmsl = missile velocity [m/sec]

The lift curve slope coefficient is computed using the equation

$$cna = 4 / b \quad (11)$$

where

$$b = sqrt(mach * mach - 1) \quad (12)$$

See Figure 6 for m_cna flow diagram.

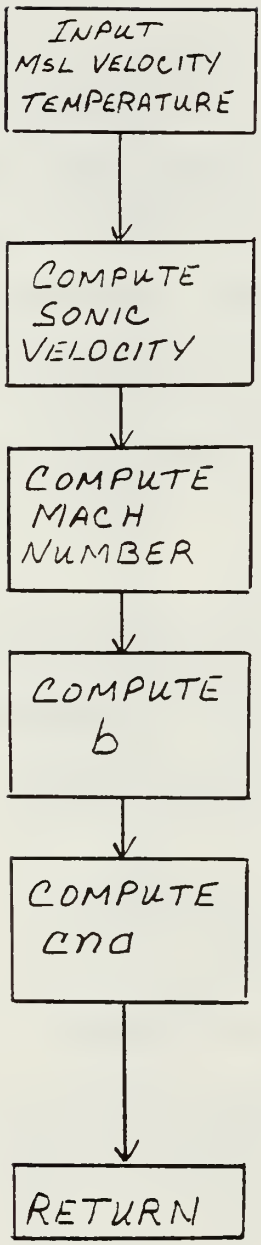

Figure 6. M_Cna Flow Diagram

3. M Guide( )

The m_guide function computes the missile command angles, alpha (missile vertical angle of attack) and beta (missile side slip angle). See Figure 7 for the m_guide flow diagram.

The MICE II Proportional Navigation guidance equations are used in computing the angles, alpha and beta, which direct the missile's flight path. The solutions for alpha and beta are base upon the difference in the target position between two consecutive time steps. For example, the closing velocity (closv) is determined by taking the difference of the range between the missile and the target from the previous time step, to the range of the current time step. Thus,

$$closv = (orng - rng) / dt \qquad (13)$$

where

closv = closing velocity [m/sec]

rng = new missile to target range [m]

orng = old missile to target range [m]

dt = time increment [sec]

The direction cosines of the line of sight vector of the new time step in the global coordinate system were computed as follows:

$$V[0] = (XTAR - XMSL)/ rng \quad \text{in X direction}$$
$$V[1] = (YTAR - YMSL)/ rng \quad \text{in Y direction} \qquad (14)$$
$$V[2] = (ZTAR - ZMSL)/ rng \quad \text{in Z direction}$$

where

XTAR, YTAR, ZTAR = target position

XMSL, YMSL, ZMSL = missile position

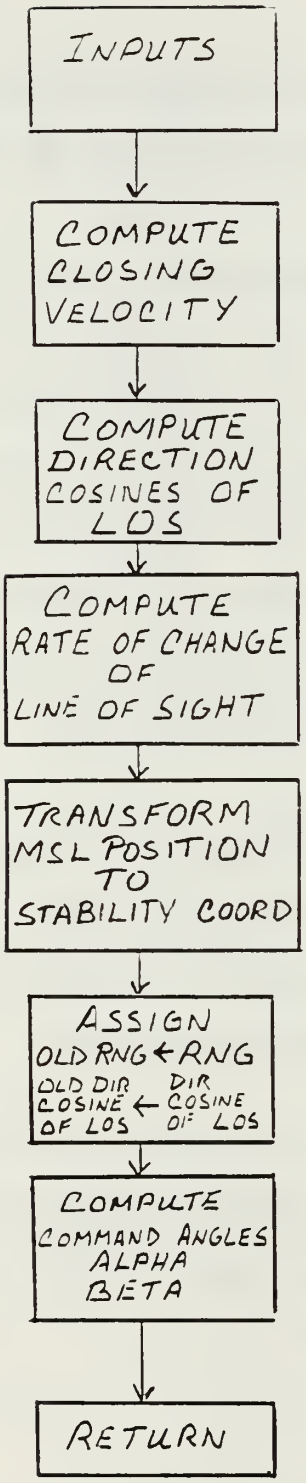

Figure 7. M_Guide Flow Diagram

The rate of change of the line of sight is computed using

$$losdt[n] = (V[n] - oV[n]) / dt \qquad (15)$$

where

losdt[n] = rate of change of the line of sight for the nth direction

V[n] = new direction cosine of line of sight

oV[n] = old direction cosine of line of sight

n = Ø,1,2

Using the tranformation matrix presented in Section C, the rate of change of the line of sight is transformed from the global coordinate system to the missile stability coordinate system.

$$m[i] = m[i] + t[i][j] * losdt[j] , \quad j = Ø,1,2 \qquad (16)$$

where

m[i] = rate of change of the line of sight along the ith direction

t[i][j] = transformation matrix elements provided in Section C

i = Ø,1,2 Ø = x direction
1 = y direction
2 = z direction

According to MICE II, alpha is given by [Ref. 1:p. 2Ø]

$$alpha = \frac{(k*closv*m[2]) * W}{cna * (S/2) * rho * vmsl * vmsl} \qquad (17)$$

where

m[2] = time rate of change of the direction cosine of the LOS vector in the z axis

W = missile weight [kg]

S = missile reference area [sq. m]

Similarly, beta is determined by,

$$beta = \frac{(k * closv * m[1]) * W}{cna * (S/2) * rho * vmsl * vmsl} \qquad (18)$$

where

m[1] = time rate of change of the direction cosine of the LOS vector in the y axis

5. M Thrust( )

The m_thrust function determines the missile thrust and missile weight as a function of time. MICE II used various numerical constants on its missile thrust and weight computations that were not thoroughly discussed. In ISEAS, missile thrust is taken as a constant value for each type of missile selected, and zero when the burn time is exceeded. The missile weight is computed using

$$W = oW - (\text{burn rate} * dt) \qquad (19)$$

where

W = new missile weight [kg]

oW = old missile weight [kg]

burn rate = missile burn rate (depends on selected missile type) [kg/sec]

dt = missile flight time increment

See Figure 8 for the m_thrust flow diagram.

6. M Flyout( )

Once the missile is fired by the user, the m_flyout function will check the following missile flight conditions at each time step:

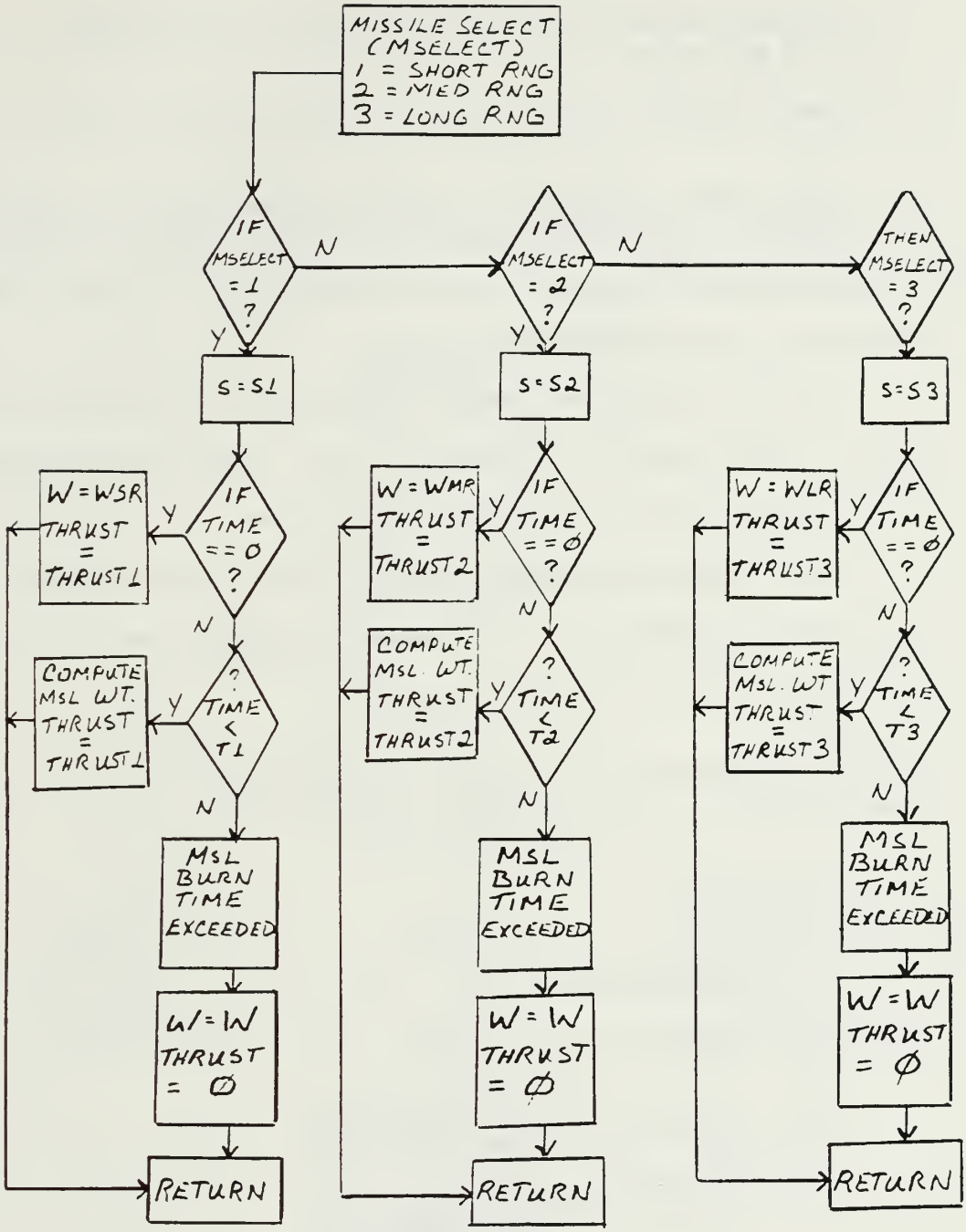

Figure 8. M_Thrust Flow Diagram

1. Missile altitude will be compared to the maximum missile altitude of 20,000 meters.

2. Time of flight will be compared to the maximum missile flight time.

3. Range between the missile and the target will be computed and compared to the maximum separation range allowed.

If any of the above flight checks are exceeded, the m_flyout function will stop and return to the main program.

Another check is the minimun range between the missile and the target. If the range is 6.1 meters or less, this will be considered a hit. See Figure 9 for the m_flyout flow diagram.

Once alpha and beta are computed by the m_guide function, lift conditions are generated and the linear acceleration equation for the vdotm is [Ref. 1:p. 6]

$$vdotm = \frac{[T - D - W * \sin(gamm) - La * alpha - Lb * beta]}{W / g} \qquad (20)$$

where

T = missile thrust [N]

D = missile drag at zero lift [N]

La = cna * q * S * alpha
missile lift due to angle alpha

Lb = cna * q * S * beta
missile lift due to angle beta

q = dynamic pressure = .5*rho*vmsl*vmsl [N/sq. m]

See Figure 10 for the forces on the missile.

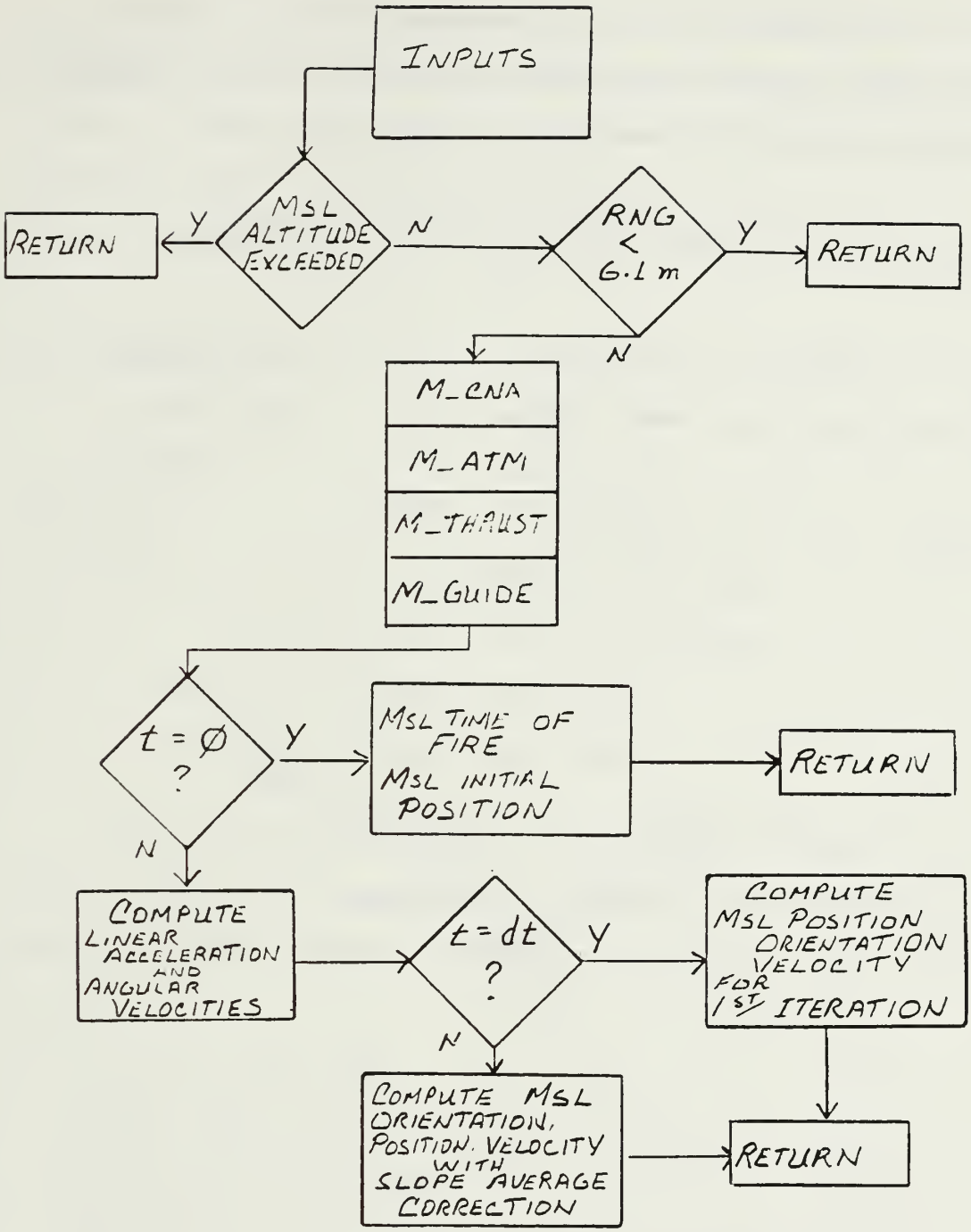

Figure 9. M_Flyout Flow Diagram

Calculation of the missile elevation angle turn rate (gamdm) due to the angle of attack, alpha, gravity and the lift component due to the net thrust is determine by using [Ref. 1:p. 6]

$$gamdm = \frac{[\,(T - D) * alpha + La - W * g * \cos(gamm)\,]}{W * vmsl} \qquad (21)$$

Similarly the missile azimuth angle turn rate (etadm) due to the side slip angle, beta, is given by [Ref. 1:p. 6]

$$etadm = \frac{[\,-(T - D) * beta - Lb * beta\,]}{W * vmsl * \cos(gamm)} \qquad (22)$$

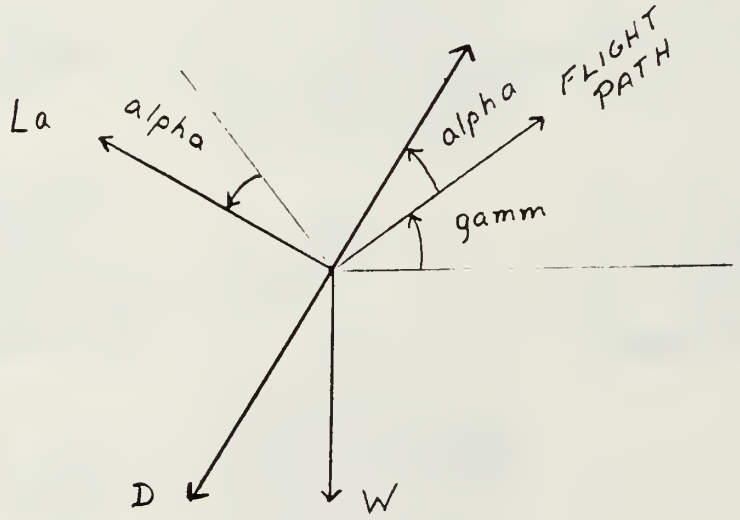

Figure 10. Forces On Missile

From the computed linear acceleration and angular velocities, an update of missile position, orientation and velocity is determined. In MICE II [Ref. 1:pp. 4-10], a slope averaging technique is used to numerically update the missile velocity and location.

The missile is at initial position at time of fire. Thus,

XMSL = XC vmsl = VC

YMSL = YC ogamm = gamm

ZMSL = ZC oetam = etam

where

XC,YC,ZC = values used in position corrections

VC = values used in velocity corrections

ogamm = old missile elevation angle

oetam = old missile azimuth angle

At the first time step (dt), the missile is in flight and computation of the linear acceleration and angular velocities provide updates to missile position, orientation and velocity. [Ref. 1:p. 9]

```
vmsl = VC + dt * vdotm                                  (23)
gamm = ogamm + dt * gamdm                               (24)
etam = oetam + dt * etadm                               (25)
XMSL = XC + .5 * dt * vmsl * cos(gamm) * cos(etam)      (26)
YMSL = YC + .5 * dt * vmsl * cos(gamm) * sin(etam)      (27)
ZMSL = ZC + .5 * dt * vmsl * sin(gamm)                  (28)
```

The following are saved after the computation of equations (23) to (28): XC, YC, ZC, gamdm, etadm, vmsl, etam, gamm, vdotm, then are used as the old values, oXC, oYC, oZC, ogamdm, oetadm, ovmsl, oetam, ogamm, ovdotm, for the next time step.

As the missile continue on its flight path, at time greater than zero, gamdm and etadm values are generated and corrections are computed by the slope average technique. [Ref. 1:p. 9]

$$GAMC = ogamm + .5 * dt * (ogamdm + gamdm) \quad (29)$$

$$ETAC = oetam + .5 * dt * (oetadm + etadm) \quad (30)$$

$$VC = oVC + .5 * dt * (ovdotm + vdotm) \quad (31)$$

$$XC = oXC + .5 * dt * (oVC + VC) * \cos(GAMC) * \cos(ETAC) \quad (32)$$

$$YC = oYC + .5 * dt * (oVC + VC) * \cos(GAMC) * \sin(ETAC) \quad (33)$$

$$ZC = oZC + .5 * dt * (oVC + VC) * \sin(GAMC) \quad (34)$$

where

GAMC = correction value for elevation angle

ETAC = correction value for azimuth angle

The missile position, orientation and velocity is updated by, [Ref. 1:p. 10]

$$vmsl = oVC + dt * vdotn \quad (35)$$

$$gamm = ogamm + dt * gamdm \quad (36)$$

$$etam = oetam + dt * etadm \quad (37)$$

$$XMSL = XC + .5 * dt * (VC + vmsl) * \cos(gamm) * \cos(etam) \quad (38)$$

$$YMSL = YC + .5 * dt * (VC + vmsl) * \cos(gamm) * \sin(etam) \quad (39)$$

$$ZMSL = ZC + .5 * dt * (VC + vmsl) * \sin(gamm) \quad (40)$$

The computed XC, YC, ZC, ETAC, GAMC, vdotm, vmsl, etadm, and gamdm are saved and becomes the old values for the next time step.

The output from this function will consist of missile position (XMSL,YMSL,ZMSL), elevation angle (gamm), azimuth angle (etam), velocity (vmsl), weight (W), and missile thrust (T). All of which the main function can utilize for graphical presentations.

F. KILL PROBABILITY

This portion will conduct the endgame. It will be developed in a future study.

IV. CONCLUSIONS AND RECOMMENDATIONS

The C programing language was used for this model because of its advertised user friendly dialogue techniques and its simple and efficient operations. Although the simplicity and generality favor the C programing language, since there is no standard definitions for the C language, compiling using Lattice C and Microsoft C compilers causes errors found from one unit but not from the other. It is therefore recommended to work with one selected compiler until the language is standardized.

MICE II simulation logics was easily followed and understood. The translation to the C language for the thrust and weight calculations presented some problems because of the numerous undocumented constant variables used in the equations. The use of the table look-up method also created difficulties in translating, therefore, alternate equations were used that provided basically the same results. All of the other major equations provided no additional difficulties in C language translation.

The use of the missile model in decision making can assist the user by providing trade-offs and relationships and by evaluatiing and comparing alternatives. But further work remains to be done. The missile model is currently working with only one guidance mode. Hopefully, additional guidance modes (command to line of sight guidance, beam rider, and track-via-missile) can be implemented.

Finally, during the early stages of this project, it is recommended that, (1) the student develope a working understanding on the mechanics of the equations of motion and guidance law, and (2) that the student complete a formal C language programing course.

LIST OF REFERENCES

1. Chan, P.T. and Huffman, R.A., Surface To Air Missile Model - Mice II, Vol I, Analyst Manual, Vought Corporation, Dallas, Texas, April 1980.

2. Lindsey, Gerald H. and Redmon, Dan R., Tactical Missile Design, notes presented for AE4703 Missile Stability and Design, Naval Postgraduate School, Monterey, California, July 1986.

3. Miller, I. and Freund, J.E., Probability And Statistics For Engineers, 3rd. Edition, Prentis-Hall, Inc., Inglewood Cliffs, New Jersey, 1985.

APPENDIX A

PROGRAM VARIABLES

The following is a listing of computer program variables in alphabetical order.

accel [m/sq.sec]	missile acceleration
alpha [rad]	missile attitude angle wrt stability axis
altmax [meters]	maximum altitude
azi [rad]	relative azimuth angle of missile launcher
beta [rad]	missile side slip angle wrt stability axis
cd	missile drag coeff
cl	missile lift coeff
closv [m/sec]	missile-target closing velocity
cna	missile lift curve slope coeff
dt [sec]	time increment
drag [N]	missile drag
elev [rad]	relative elevation angle of missile launcher
etam [rad]	azimuth angle of missile velocity vector
etac	correction value for azimuth angle
etadm [rad/sec]	rate of change of missile azimuth angle
g[kg-m/N-sq.sec]	gravity
gamc	correction value for elevation angle
gamm [rad]	elevation angle of missile velocity vector
gamdm [rad/sec]	rate of change of missile elevation angle
ko	base value of Proportional Navigation constant
lift [N]	missile lift
losdt [rad/sec]	rate of change of LOS
m[] [m]	missile position in stability coord
mach	missile mach number
mbrl [kg/sec]	long range missile burn rate
mbrm [kg/sec]	med range missile burn rate
mbrs [kg/sec]	short range missile burn rate
m_lr	long range missile (3)
m_mr	med range missile (2)
mselect	missile select by user
m_sr	short range missile

oetadm	old rate of change azimuth angle
oetam	old azimuth angle
ogamdm	old rate of change elevation angle
ogamm	old elevation angle
orng [meters]	old range between missile and target
ov[] [rad]	old direction cosine of LOS
ovc	old correction value for missle velocity
oxc	old correction value for xmsl
oyc	old correction value for ymsl
ozc	old correction value for zmsl
q	dynamic pressure
rho [kg/cu.m]	atmospheric density
rng [meters]	range between missile and target
rng1 [m]	max range of short range missile
rng2 [m]	max range of med range missile
rng3 [m]	max range of long range missile
rtint [m]	target intercept range
s [sq. meters]	surface area of missile
t [secs]	missile flight time
t1 [secs]	max flight time for short range missile
t2 [secs]	max flight time for med range missile
t3 [secs]	max flight time for long range missile
temp [kelvin]	atmospheric temperature
thrust [N]	missile thrust
thrust1 [N]	thrust of short range missile
thrust2 [N]	thrust of med range missile
thrust3 [N]	thrust of long range missile
v[] [rad]	direction cosine of LOS
vc	correction value for vmsl
vmsl [m/s]	missile velocity
w [kg]	missile weight
wlr [kg]	weight of long range missile
wmr [kg]	weight of med range missile
wsr [kg]	weight of short range missile
xc	correction value for xmsl
xmsl [m]	missile position on x axis
xtar [m]	target position on x axis
xtint [m]	target intercept position on the x axis
yc	correction value for ymsl
ymsl [m]	missile position on y axis
ytar [m]	target position on y axis
ytint [m]	target intercept position on the y axis

zc	correction value for zmsl
zmsl [m]	missile position on z axis (missile altitude)
ztar [m]	target position on z axis
ztint [m]	target intercept position on the z axis

APPENDIX B

PROGRAM LISTINGS

The following is a "C" language programming listing of the ISEAS missile simulation program.

```
m_lchr()
/* This function will compute the launcher's azimuth and elevation
   angles with respect to the ship's heading.  Input to this
   function will the launcher (the initial missile position)
   position and the calculated target intercept position.  The
   output will be the launcher's azimuth and elevation angles
   and must be converted to the global coordinates and becomes
   the initial gamm and etam for m_flyout.                    */

{

/* COORDINATES ARE RELATIVE TO SHIP'S HEAD */
  float xtint,ytint,ztint; /*tgt intercept position*/
  float xtar,ytar,ztar;   /*tgt rel position*/
  float xmsl,ymsl,zmsl; /*launcher or initial missile position*/
  float elmts=90.;           /*max lchr elevation*/
  float almts=135.;  /*lchr port/stbd bearing limits*/
  float rng;      /* range to target */
  float rtint /* intercept range */
  float rng1=20000,rng2=48000,rng3=121000; /* missile ranges */
  float elev, azi;
  float x,y,z,xy;

/**** COMPUTE LAUNCHER BEARING ANGLE *********/
  azi = atan((ytint-ymsl)/(xtint-xmsl));
  azi = (180*azi)/3.14159;
  if (abs(azi) > almts)
     {
      printf("\n BEARING ANGLE LIMITS EXCEEDED");
      return;
     }

/**** COMPUTE LAUNCHER ELEVATION ANGLE ******/
  x = xtint-xsml;
  y = ytint-ysml;
  z = ztint-zsml
  xy = sqrt(x*x + y*y);
  elev = atan(z/xy);
  elev = (180*elev)/3.14159;
  if (elev > elmts)
     {
      printf("\n ELEVATION LIMITS EXCEEDED");
      return;
     }

/**** CHECK IF WITHIN MISSILE RANGE ******/
      rtint = sqrt(x*x + y*y + z*z);
      if ( mselect == 1 )
         {
```

```
        if ( rtint > rng1 )
           {
            printf("\nTARGET NOT WITHIN SEEKER RANGE");
            printf("\nCONTINUE TRACKING");
            return;
           }
        }
     if ( mselect == 2 )
        {
         if ( rtint > rng2 )
           {
            printf("\nTARGET NOT WITHIN SEEKER RANGE");
            printf("\nCONTINUE TRACKING");
            return;
           }
         }
     if ( mselect == 3 )
        {
         if ( rtint > rng3 )
           {
            printf("\nTARGET NOT WITHIN SEEKER RANGE");
            printf("\nCONTINUE TRACKING");
            return;
            }
        }
```

```
/**** PRINT LAUNCHER AZIMUTH AND ELEVATION ****/
  if ( azi < 0 )

      printf("\n LAUNCHER BEARING = %.2f DEG TO PORT",azi);
    else
      printf("\n LAUNCHER BEARING = %.2f DEG TO STBD",azi);

   printf("\n LAUNCHER ELEV = %.2f DEG",elev);

   printf("\n TARGET WITHIN RANGE");

   printf("\n LAUNCHER READY");


  }
```

```
#include<stdio.h>
#include<math.h>

/* This program was created to simulate the main function and
   provide target data to evaluate variable inputs and outputs
   of the flyout function.  Other functions are also called upon
   to provide the necessary values for the computations of the
   missile's equations of motion and eventually, the  missile's
   position in global coordinates.                              */

/* GLOBAL VARIABLES */
  float xtar=10000,ytar=10000,ztar=10000;/*current target coord*/
  float xmsl,ymsl,zmsl;/*current missile coord */
  float vmsl=671;   /* missile velocity = 2200 ft/sec */
  float gamm,ogamm;      /* elev angle of vmsl */
  float gamdm,ogamdm;
  float etadm,oetadm;
  float etam,oetam;      /* azi angle of vmsl */
  float w;             /* msl weight */
  float s;            /* area of selected msl */
  float s1=2.33;      /* ref area of short rng msl */
  float s2=4.92;     /* ref area of med rng msl */
  float s3=8.78;    /* ref area of long rng msl */
  float thrust;     /* msl thrust [ N ] */
  float drag,cd=0.2;   /* msl drag,drag coeff */
  float lift,cl;    /* msl lift,lift coeff */
  float dt;         /* time increment delta-t */
  float altmax=21000;      /* msl maximum altitude in meters */
  float t;          /* msl flight time */
  float rng;
  float orng=20000; /* initial value from main function */
  float rho,temp,g=1;
  float alpha,beta;
  float cna;
  float ko;    /* input base value of proportional constant*/
  int mselect;
  float v[3],ov[3];
  float cpa;
  int shoot;
  int ans,ans1,ans2,ans3;
  float vc,xc,yc,zc;
  float ovc,oxc,oyc,ozc;
  float vdotm,ovdotm;
  float gamc,etac,ogamc,oetac;
  float wsr=231.3,wmr=581,wlr=1066;
  float mbrs=1.36,mbrm=0.338,mbrl=0.338;
  float t1=30,t2=71,t3=181;
  float thrust1=200000,thrust2=400000,thrust3=400000;
```

```
main()
  {
       printf("\n Change target position? 1=yes/Ø=no  ");
       scanf("%d", &ans);
       if ( ans == 1 )
          {
            printf("\nEnter target position in meters");
            printf("\n xtar =  ");
            scanf("%f", &xtar);
            printf("\n ytar =  ");
            scanf("%f", &ytar);
            printf("\n ztar =  ");
            scanf("%f", &ztar);

           }
       printf("\n Initial target position in meters");
       printf("\n xtar = %f ytar = %f ztar = %f",xtar,ytar,ztar);
       printf("\n Target heading is fixed at 27Ø degs");
       printf("\n at 1Ø meters decrement per time step");

       printf("\n\n Enter missile initial position in meters");
       printf("\n xmsl =  ");
       scanf("%f", &xmsl);
       printf("\n ymsl =  ");
       scanf("%f", &ymsl);
       printf("\n zmsl =  ");
       scanf("%f", &zmsl);

       printf("\n Enter missile attitude angle in radians =  ");
       scanf("%f", &gamm);
       printf("\n Enter missile azimuth angle in radians =  ");
       scanf("%f", &etam);

       printf("\n Desired time increment in secs =   ");
       scanf("%f", &dt);

       printf("\n Enter proportional constant base value =   ");
       scanf("%f", &ko);

       printf("\n Select type of missile ");
       printf("\nShort Range= 1  Med Range= 2  Long Range= 3 ");
       printf("\n missile select =  ");
       scanf("%d", &mselect);

       printf("\n Change missile ref area? 1=yes/Ø=no   ");
       scanf("%f", &ans1);
       if ( ans1 == 1 )
          {
```

```
    if (mselect == 1)
       {
        printf("\n Enter: s1 =   ");
        scanf("%f", &s1);
        }
    if (mselect == 2)
       {
        printf("\n Enter: s2 =  ");
        scanf("%f", &s2);
        }
    if (mselect == 3)
       {
        printf("\n Enter: s3 =  ");
        scanf("%f", &s3);
        }
   }

printf("\n Change missile thrust? 1=yes/Ø=no    ");
scanf("%d", &ans2);
if ( ans2 == 1 );
   {
    if ( mselect == 1 )
       {
        printf("\n Enter: thrust1 =   ");
        scanf("%f", &thrust1);
       }
    if ( mselect == 2 )
       {
        printf("\n Enter: thrust2 =  ");
        scanf("%f", &thrust2);
       }
    if ( mselect == 3 )
       {
        printf("\n Enter: thrust3 =  ");
        scanf("%f", &thrust3);
       }
   }

printf("\n Change missile burn rate? 1=yes/Ø=no  ");
scanf("%d", &ans3);
if ( ans3 == 1 )
   {
     if ( mselect == 1 )
        {
         printf("\n Enter: burn rate 1 =  ");
         scanf("%f", &mbrs);
         }
```

```
            if ( mselect == 2 )
               {
                printf("\n Enter: burn rate 2 =  ");
                scanf("%f", &mbrm);
               }
            if ( mselect == 3 )
               {
                printf("\n Enter: burn rate 3 =  ");
                scanf("%f", &mbrl);
               }
         }


   printf("\n When ready to shoot ");
   spacebar();
   t=Ø;

   while ( t < 31 )
       {
        printf("\n\n t = %f",t);
        m_flyout();

        xtar = xtar - 1Ø;
        printf("\n xtar = %f",xtar);
        ytar = ytar;
        printf("  ytar = %f",ytar);
        ztar = ztar;
        printf("  ztar = %f",ztar);
        t = t + dt;
       }
  printf("\n End of flight ");
  }



spacebar()
{
  printf("\n%s"," press spacebar");
  while ( getch() !=' ');
}
```

```
m_flyout()
/* This function computes the missile's linear accelerations
   and angular velocities.  Inputs of missile and target
   positions, and missile general characteristics are required.
   Computations of missile attitude and azimuth, velocity,
   weight, thrust, and position are the outputs and are updated
   at each time increment                                        */
 {
  float a,b;
  float x,y,z;

  x = xtar - xmsl;
  y = ytar - ymsl;
  z = ztar - zmsl;

/*************** range between missile and target **************/

  rng = sqrt( x*x + y*y + z*z);
  printf("\n range = %f meters",rng);

/********** altitude check of missile flight ******************/

  if ( zmsl > altmax )
    {
     printf("\n max altitude exceeded ");
     return;
     }


/********** CPA check of missile flight **********************/

  if (rng <= 6.1 )  /*cpa .ØØ61 km or less */
     return;  /* warhead exploded- calc miss dist */

  if (rng > orng)
    {
     printf("\n missile pass CPA");
     cpa = orng;
     printf("\n CPA = %f m ",cpa);
     t = 31;
     return;
     }


/********* call function to calc temp and density wrt   ********/
/********* altitude - return value of temp and density *********/

  m_atm();
  printf("\n At an altitude of %f",zmsl);
  printf("\n temperature = %f",temp);
  printf("\n density = %f",rho);
```

```
/********* call m_cna function to calc the lift curve slope   **/
/********* coefficient ( cna )  with respect to altitude and  **/
/********* temperature                                        **/

   m_cna();
   printf("\n cna = %f",cna);

/******** call function to calc msl propulsion forces   ********/
/******** return with thrust and msl weight ********************/

  m_thrust();


/******* call function to calc msl guidance - return ***********/
/******* alpha and beta - msl command angles ******************/

  m_guide();

/********** calc drag at zero lift ****************************/

  printf("\n cd = %f",cd);
  printf("\n rho = %f",rho);
  printf("\n s = %f",s);
  printf("\n vmsl = %f",vmsl);
  drag = cd*(s/2)*rho*vmsl*vmsl;
  printf("\n drag = %f",drag);

/********** At time zero, missile is fire and initial**********/
/********** misssile position is launcher position ************/
  if ( t == 0 )
     {
      printf("\n At time of fire");
      printf("\n xmsl= %f  ymsl= %f  zmsl= %f",xmsl,ymsl,zmsl);
      printf("\n etam= %f  gamm= %f",etam,gamm);
      vc = vmsl;
      xc = xmsl;
      yc = ymsl;        /** values assigned to old values **/
      zc = zmsl;        /** for the next time step        **/
      oetam = etam;
      ogamm = gamm;
      return;
     }

/************* calc of msl linear accelaration ****************/

  a = cna*.5*s*rho*vmsl*vmsl*alpha;
  b = cna*.5*s*rho*vmsl*vmsl*beta;
  printf("\n a = %f   b= %f ",a,b);
  vdotm=((thrust-drag-(w*g*sin(gamm))-(a*alpha+b*beta))/w);
  printf("\n vdotm = %f",vdotm);
```

```
/************ calc of msl elavation angle turn rate ************/

  gamdm=((thrust-drag)*alpha+a*w*g-(w*g*cos(gamm)))/(w*vmsl);
  printf("\n gamdm = %f",gamdm);

/************ calc of msl azimuth angle turn rate **************/

  etadm=(-(thrust-drag)*beta-b)/(w*vmsl*cos(gamm));
  printf("\n etadm = %f",etadm);


/**** At this time, the missile is in flight and if the time **/
/**** step equals the time of increment, then first iteration**/
/**** of missile position, orientation, and velocity is     ***/
/**** computed.                                             ***/

  if ( t == dt )
   {

    /* update of msl vel */
    vmsl = vc + Ø.5*dt*vdotm;

    /* new msl elevation angle */
    gamm = ogamm + Ø.5*dt*gamdm;

    /* new msl azimuth angle */
    etam = oetam + Ø.5*dt*etadm;

    /* update msl position base on flight correction */

    xmsl = xc + Ø.5*vmsl*dt*cos(gamm)*cos(etam);

    ymsl = yc + Ø.5*vmsl*dt*cos(gamm)*sin(etam);

    zmsl = zc + Ø.5*vmsl*dt*sin(gamm);

    /* change new value to old value for next iteration */

    ovdotm = vdotm;
    ogamdm = gamdm;
    oetadm = etadm;
    ovc = vmsl;
    oxc = xc;
    oyc = yc;
    ozc = zc;
    oetam = etam;
    ogamm = gamm;
   }
```

```
/**** Since the time step is greater than the time increment **/
/**** and greater than zero, the missile is well into its    **/
/**** flight path and the missile position, orientation, and **/
/**** velocity are computed, and are corrected by the slope  **/
/**** average technique                                      **/
  else
    {
     gamc = ogamm + .5 * dt * (ogamdm + gamdm);
     etac = oetam + .5 * dt * (oetadm + etadm);
     vc = ovc + .5 * dt * (ovdotm + vdotm);
     xc = oxc + .5 * dt * (ovc + vc) * cos(gamc) * cos(etac);
     yc = oyc + .5 * dt * (ovc + vc) * cos(gamc) * sin(etac);
     zc = ozc + .5 * dt * (ovc + vc) * sin(gamc);

     vmsl = vc + dt * vdotm;
     gamm = ogamm + dt * gamdm;
     etam = oetam + dt * etadm;
     xmsl = xc + .5 * dt * (vc + vmsl) * cos(gamm) * cos(etam);
     ymsl = yc + .5 * dt * (vc + vmsl) * cos(gamm) * sin(etam);
     zmsl = zc + .5 * dt * (vc + vmsl) * sin(etam);

     ovdotm = vdotm;
     ogamdm = gamdm;
     oetadm = etadm;
     oetam = etac;
     ogamm = gamc;
       ovc = vc;
       oxc = xc;
       oyc = yc;
       ozc = zc;
    }

/********* Print missile position, orientation, and velocity **/
/********* as well as missile thrust and weight               **/
  printf("\nxmsl= %f m  ymsl= %f m  zmsl= %f m ",xmsl,ymsl,zmsl);
  printf("\n vmsl = %f m/sec",vmsl);
  printf("\n gamm = %f  etam = %f ",gamm,etam);
  printf("\n w = %f kg   thrust = %f N",w,thrust);
 return;
  }
```

```
m_atm( )
/* This function will calculate the atmospheric density and
   temperature with respect to altitude.  Equations used were
   derived by the linear regression method.  Maximum missile
   altitude is 21000 meters, and the flyout subroutine will
   check if the missile exceeds maximum altitude.  Therefore as
   long as the missile is at 21000 meters or below, atmospheric
   density and temperature will be computed.                    */

  {

/******  density computation  [ kg / cu. m ]  *****************/

    rho = ( 1.0177572 - (0.000048539 * zmsl) );



/******  temperature computation  [ kelvin ]  *****************/

    if ( zmsl >= 11500 )   /** at this altitude, temperature **/
        temp = 216.66;     /** is constant at 216.66 k       **/
     else
        temp = 287.25 - (0.006241865 * zmsl);


/** Return to flyout subroutine with values of temperature and
    density                                                    **/

   }
```

```
m_cna()
/* This function calculates the lift curve slope
   coefficeint with respect to altitude and temperature.  CNA
   is used for the computation of alpha, beta, and the equations
   of motions.                                                    */
  {
    float accel;
    float mach;
    float b;

/**** calc missile accelaration  ****/

   accel = 2Ø.Ø5 * sqrt(temp);


/**** calc missile mach number   ****/

   mach =  vmsl / accel;


   b = sqrt((mach*mach) - 1);


/**** calc lift curve slope coeff   ****/

   cna = 4 / b;

   return;

  }
```

```
m_thrust()
/* The m_thrust function computes the missile weight during
   flight.  Missile thrust is a constant value and this value
   will depend on the type of missile selected.  At burn out,
   missile thrust will be zero and flight time is exceeded.
   Output from this function will be missile weight and missile
   thrust.                                                    */
  {

/*** if user selects a short range missile  ***/

    if ( mselect == 1 )
       {
         s = s1;
         if ( t == Ø )  /** at missile time of fire  **/
            {
              w = wsr;
              thrust = thrust1;
              return;
            }
         else
            {
              if ( t < t1 )  /** missile flight time  **/
                 {
                   w = w - (mbrs * dt);
                   thrust = thrust1;
                   return;
                 }
              else
                 {
                   printf("\n missile burn time compeleted");
                   w = w;
                   thrust = Ø;
                   return;
                 }
            }
       }

/*** user selects a medium range missile   ***/

  if ( mselect == 2 )
     {
       s = s2;
       if ( t == Ø )  /** missile time of fire  **/
          {
            w = wmr;
            thrust = thrust2;
            return;
          }
```

```
        else
           {
             if ( t < t2 )  /** missile flight time **/
                {
                  w = w - (mbrm * dt);
                  thrust = thrust2;
                  return;
                }
             else
                {
                  printf("\n missile burn time completed");
                  w = w;
                  thrust = Ø;
                }
           }
      }


/***   if user select a long range missile   ***/

  if ( mselect == 3 )
     {
       s = s3;
       if ( t == Ø )   /*** missile time of fire  ***/
          {
            w = wlr;
            thrust = thrust3;
            return;
          }
       else
          {
            if ( t < t3 )  /*** missile time of flight ***/
               {
                 w = w - (mbrl * dt);
                 thrust = thrust3;
                 return;
               }

            else
               {
                 printf("\n missile burn time completed");
                 w = w;
                 thrust = Ø;
                 return;
               }
          }
     }
  }
```

```
m_guide()
/* This function computes the missile command angles, alpha
   (missile angle of attack) and beta (missile side slip angle),
   during a proportional navigation flight path.  The command
   angles are with respect to the missile stability and are the
   output of this function.                                    */
  {
   int c;
   int i,j;
   float m[3];
   float t[3][3];
   float losdt[3];
   float closv;
   float x,y,z;


   x = xtar - xmsl;
   y = ytar - ymsl;
   z = ztar - zmsl;

   rng = sqrt( x*x + y*y + z*z );

   closv = ( orng - rng )/ dt;

/*** calc direction cosine of LOS in global coordinate  ***/

   v[0] = x/rng;
   v[1] = y/rng;
   v[2] = z/rng;

/** calc the rate of change of LOS between missile and target **/

   for ( c = 0; c < 3; ++c )
      {
       losdt[c] = ( v[c] - ov[c] )/ dt;
       printf("\n losdt[%d] = %f",c,losdt[c]);
       }

/***  matrix elements  ***/

   t[0][0] = cos(gamm) * cos(etam);
   t[0][1] = cos(gamm) * sin(etam);
   t[0][2] = sin(gamm);
   t[1][0] = sin(etam);
   t[1][1] = -cos(etam);
   t[1][2] = 0;
   t[2][0] = -sin(gamm) * cos(etam);
   t[2][1] = -sin(gamm) * sin(etam);
   t[2][2] = cos(gamm);
```

```
/*** transform missile position from global coordinates to
    missile stability coordinates                          ***/
   for ( i = 0; i < 3; ++i )
      {
       m[i] = 0;
       for ( j = 0; j < 3; ++j)
           m[i] = m[i] + t[i][j]*losdt[j];
       }

/*** new direction cosine of LOS converted to old direction
     cosine of LOS for the next iteration.                 ***/

    ov[0] = v[0];
    ov[1] = v[1];
    ov[2] = v[2];

/*** computed range between missile and target is now the
     old range for the next iteration.                     ***/

    orng = rng;
    printf("\n old range = %f meters",orng);
/*** calc the missile command angles, alpha and beta.     ***/

    alpha = ((ko*closv*m[2])*w) / (0.5*cna*s*rho*vmsl*vmsl);

    beta = (ko*closv*m[1])*w / (0.5*cna*s*rho*vmsl*vmsl);

    printf("\n alpha = %f",alpha);
    printf("\n beta = %f",beta);

    return;

   }
```

BIBLIOGRAPHY

Clemow, J., Missile Guidance, Richard Clay and Company, Ltd., Great Britain, 1962.

Frieden, D. R., ed., Principles Of Naval Weapons Systems, Naval Institute Press, Annapolis, Maryland, 1985.

Paarmann, L. D., Faraone, J. N., and Smoots, C. W., Guidance Law Handbook For Classical Proportional Navigation, ITT Research Institute, June 1978.

Merril, G., ed., Principles Of Guided Missile Design, D. Van Nostramd Co., Inc., Princeton, New Jersey, 1955.

Myers, C. T., Guided Missiles, Operations, Design And Theory, McGraw - Hill Book Co., Inc., New York, 1958.

Rose, R. H. and Dloogatch, M. A., Generic Surface To Air Missile Model, Vol 1, Analyst Manual, A. T. Kearney, Inc., Chicago, Illinois, October 1981.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Department Chairman, Code 67 Department of Aeronautics Naval Postgraduate School Monterey, California 93943	1
4.	Professor Robert E. Ball, Code 67Bp Department of Aeronautics Naval Postgraduate School Monterey, California 93943	1

220914

Thesis
A5656 Antonio
c.1 A missile flyout model
for ISEAS.